Casos Prácticos de Estudio para Diseño de Bases de Datos en MySQL

Estructura Propuesta

Fase	Contenido	Herramienta
1	**Introducción a MySQL y MySQL Workbench**	MySQL Workbench
2	**DDL práctico**: Crear base de datos "universidad"	Editor SQL
3	**DML práctico**: Poblar tablas y consultas básicas	Editor SQL
4	**JOIN simple**: Relacionar 2-3 tablas	Editor SQL
5	**Ejercicios de consolidación**	MySQL Workbench
6	**Cierre**: Buenas prácticas y preguntas	Discusión

Caso de Estudio: Sistema de Gestión Universitaria Simple

Contexto: Universidad que necesita gestionar:

- **Facultades** (id, nombre, decano)
- **Carreras** (id, nombre, id_facultad, duración_semestres)
- **Estudiantes** (id, código, nombres, apellidos, id_carrera, fecha_ingreso)
- **Profesores** (id, código, nombres, apellidos, id_facultad, especialidad)
- **Materias** (id, código, nombre, id_carrera, creditos)
- **Inscripciones** (id, id_estudiante, id_materia, semestre, año, nota_final)

Relaciones simples:

- Una facultad tiene muchas carreras
- Una carrera tiene muchos estudiantes y muchas materias
- Un estudiante se inscribe en muchas materias
- Un profesor pertenece a una facultad

Fase 1: Introducción a MySQL

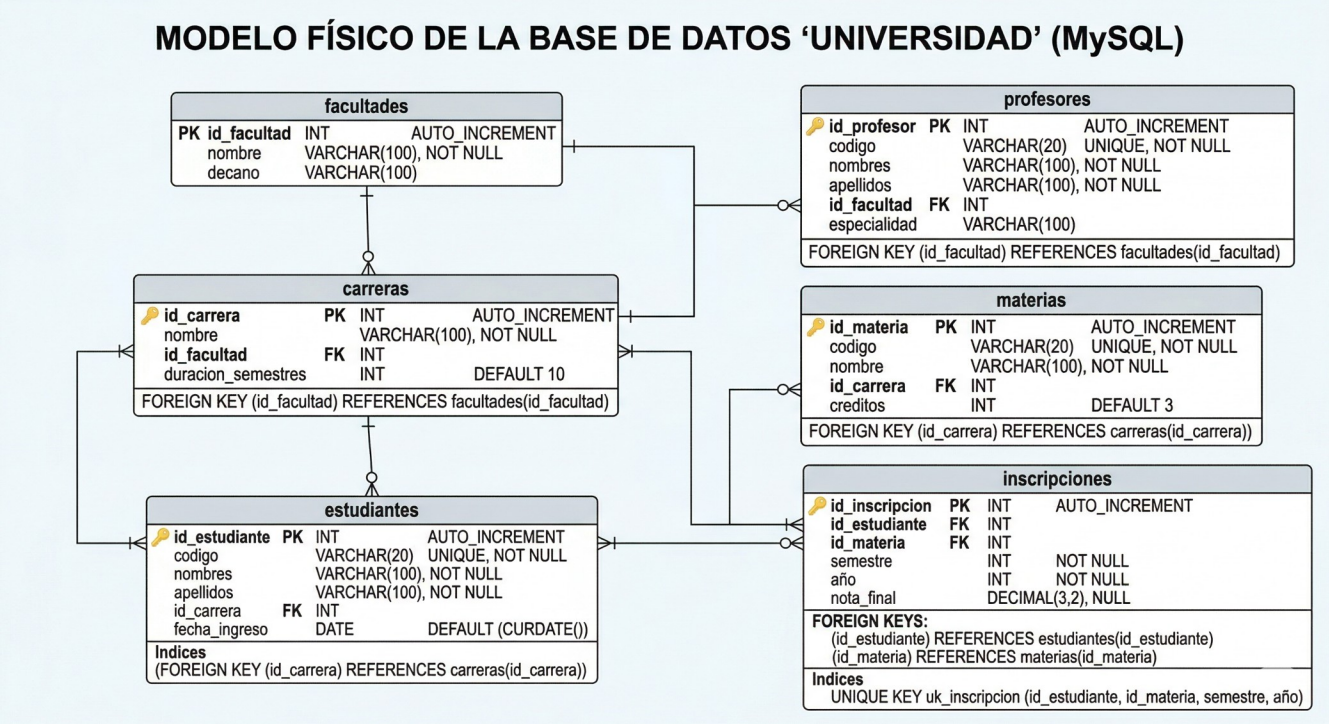
Objetivos:

- Conectar a MySQL Workbench
- Crear una conexión al servidor local
- Conocer la interfaz: Navigator, Query Editor, Result Grid

Actividad guiada:

1. Abrir MySQL Workbench
2. Crear conexión "localhost" (root / contraseña)
3. Explorar el esquema "sys" o "mysql" (solo ver, no tocar)
4. Crear un nuevo esquema llamado `universidad_practica`

Fase 2: DDL - Crear la base de datos



Script completo :
sql

```
-- Crear base de datos
CREATE DATABASE IF NOT EXISTS universidad;
USE universidad;

-- Tabla facultades
CREATE TABLE facultades (
    id_facultad INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    decano VARCHAR(100)
);

-- Tabla carreras
CREATE TABLE carreras (
```

```
    id_carrera INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    id_facultad INT,
    duracion_semestres INT DEFAULT 10,

    FOREIGN KEY (id_facultad) REFERENCES facultades(id_facultad)
);

-- Tabla estudiantes
CREATE TABLE estudiantes (
    id_estudiante INT AUTO_INCREMENT PRIMARY KEY,
    codigo VARCHAR(20) UNIQUE NOT NULL,
    nombres VARCHAR(100) NOT NULL,
    apellidos VARCHAR(100) NOT NULL,
    id_carrera INT,
    fecha_ingreso DATE DEFAULT (CURDATE()),

    FOREIGN KEY (id_carrera) REFERENCES carreras(id_carrera)
);

-- Tabla profesores
CREATE TABLE profesores (
    id_profesor INT AUTO_INCREMENT PRIMARY KEY,
    codigo VARCHAR(20) UNIQUE NOT NULL,
    nombres VARCHAR(100) NOT NULL,
    apellidos VARCHAR(100) NOT NULL,
    id_facultad INT,
    especialidad VARCHAR(100),

    FOREIGN KEY (id_facultad) REFERENCES facultades(id_facultad)
);

-- Tabla materias
CREATE TABLE materias (
```

```
    id_materia INT AUTO_INCREMENT PRIMARY KEY,
    codigo VARCHAR(20) UNIQUE NOT NULL,
    nombre VARCHAR(100) NOT NULL,
    id_carrera INT,
    creditos INT DEFAULT 3,

    FOREIGN KEY (id_carrera) REFERENCES carreras(id_carrera)
);

-- Tabla inscripciones (tabla de relación muchos a muchos)
CREATE TABLE inscripciones (
    id_inscripcion INT AUTO_INCREMENT PRIMARY KEY,
    id_estudiante INT,
    id_materia INT,
    semestre INT NOT NULL,
    año INT NOT NULL,
    nota_final DECIMAL(3,2) NULL, -- 0.00 a 5.00

    FOREIGN KEY (id_estudiante) REFERENCES estudiantes(id_estudiante),
    FOREIGN KEY (id_materia) REFERENCES materias(id_materia),

    -- Evitar inscripción duplicada
    UNIQUE KEY uk_inscripcion (id_estudiante, id_materia, semestre, año)
);
```

Ejercicios durante la fase:

- Ejecutar el script completo
- Verificar con `SHOW TABLES;`
- Describir una tabla: `DESCRIBE estudiantes;`
- Modificar una tabla: agregar columna email a estudiantes

Fase 3: DML - Poblar y consultar

Inserción de datos:
sql

```
-- Facultades
```

```
INSERT INTO facultades (nombre, decano) VALUES
('Ingeniería', 'Dr. Carlos Martínez'),
('Ciencias Básicas', 'Dra. Ana López'),
('Ciencias Sociales', 'Dr. Pedro Gómez');

-- Carreras
INSERT INTO carreras (nombre, id_facultad, duracion_semestres) VALUES
('Ingeniería de Sistemas', 1, 10),
('Tecnología en Desarrollo de Software', 1, 6),
('Matemáticas', 2, 10),
('Psicología', 3, 10);

-- Estudiantes
INSERT INTO estudiantes (codigo, nombres, apellidos, id_carrera, fecha_ingreso)
VALUES
('202310001', 'Juan Carlos', 'Rodríguez Pérez', 2, '2023-02-01'),
('202310002', 'María Fernanda', 'López García', 2, '2023-02-01'),
('202310003', 'Carlos Alberto', 'Martínez Silva', 1, '2022-02-01'),
('202320001', 'Ana Lucía', 'Gómez Torres', 3, '2023-08-01');

-- Profesores
INSERT INTO profesores (codigo, nombres, apellidos, id_facultad, especialidad)
VALUES
('P001', 'Luis Eduardo', 'Fernández Ruiz', 1, 'Bases de Datos'),
('P002', 'Diana Patricia', 'Castro Mendoza', 1, 'Programación Web'),
('P003', 'Roberto', 'Gutiérrez Vargas', 2, 'Estadística');

-- Materias
INSERT INTO materias (codigo, nombre, id_carrera, creditos) VALUES
('BD101', 'Base de Datos I', 2, 3),
('BD201', 'Base de Datos II', 2, 3),
('PG101', 'Programación I', 2, 4),
('PG102', 'Programación II', 1, 4),
('MT101', 'Cálculo I', 3, 4);
```

```
-- Inscripciones
INSERT INTO inscripciones (id_estudiante, id_materia, semestre, año, nota_final)
VALUES
(1, 1, 1, 2023, 4.5),  -- Juan Carlos en BD I, nota 4.5
(1, 3, 1, 2023, 4.0),  -- Juan Carlos en Programación I
(2, 1, 1, 2023, 3.8),  -- María en BD I
(2, 3, 1, 2023, 4.2),  -- María en Programación I
(3, 4, 4, 2023, 4.7);  -- Carlos en Programación II
```

Consultas básicas:

sql

```
-- 1. Listar todos los estudiantes
SELECT * FROM estudiantes;

-- 2. Estudiantes de una carrera específica
SELECT e.codigo, e.nombres, e.apellidos, c.nombre AS carrera
FROM estudiantes e
JOIN carreras c ON e.id_carrera = c.id_carrera
WHERE c.nombre = 'Tecnología en Desarrollo de Software';

-- 3. Promedio de notas por estudiante
SELECT
    e.codigo,
    CONCAT(e.nombres, ' ', e.apellidos) AS estudiante,
    AVG(i.nota_final) AS promedio,
    COUNT(i.id_materia) AS materias_inscritas
FROM estudiantes e
JOIN inscripciones i ON e.id_estudiante = i.id_estudiante
GROUP BY e.id_estudiante;

-- 4. Materias sin estudiantes inscritos (LEFT JOIN)
SELECT m.codigo, m.nombre, c.nombre AS carrera
FROM materias m
```

```
JOIN carreras c ON m.id_carrera = c.id_carrera
LEFT JOIN inscripciones i ON m.id_materia = i.id_materia
WHERE i.id_inscripcion IS NULL;

-- 5. Actualizar nota
UPDATE inscripciones
SET nota_final = 4.8
WHERE id_estudiante = 1 AND id_materia = 1;

-- 6. Eliminar inscripción (ejemplo académico)
DELETE FROM inscripciones
WHERE id_estudiante = 2 AND id_materia = 3;
```

Fase 4: JOINs - Uniendo tablas

Ejercicios progresivos:
sql

```
-- Nivel 1: INNER JOIN simple (solo registros que existen en ambas tablas)
SELECT
    e.nombres,
    e.apellidos,
    c.nombre AS carrera,
    f.nombre AS facultad
FROM estudiantes e
INNER JOIN carreras c ON e.id_carrera = c.id_carrera
INNER JOIN facultades f ON c.id_facultad = f.id_facultad;

-- Nivel 2: LEFT JOIN (todos los estudiantes, incluso sin carrera)
COALESCE es una función que devuelve el primer valor no nulo de una lista de
argumentos.

SELECT
    e.nombres,
    e.apellidos,
```

```
    COALESCE(c.nombre, 'Sin carrera asignada') AS carrera
FROM estudiantes e
LEFT JOIN carreras c ON e.id_carrera = c.id_carrera;

-- Nivel 3: Tres tablas con agregación
SELECT
    f.nombre AS facultad,
    COUNT(DISTINCT c.id_carrera) AS total_carreras,
    COUNT(DISTINCT e.id_estudiante) AS total_estudiantes
FROM facultades f
LEFT JOIN carreras c ON f.id_facultad = c.id_facultad
LEFT JOIN estudiantes e ON c.id_carrera = e.id_carrera
GROUP BY f.id_facultad;
```

Fase 5: Ejercicios de consolidación

Ejercicios para resolver en clase:

1. Agregar 2 nuevas materias a la carrera de Tecnología en Desarrollo de Software
2. Inscribir al estudiante Carlos Alberto en Base de Datos I con nota 4.2
3. Listar todos los profesores de la facultad de Ingeniería
4. Calcular cuántos estudiantes ingresaron en 2023
5. Mostrar la carrera con más estudiantes

Fase 6: Cierre

Checklist de verificación:

- [ ] Base de datos creada sin errores
- [ ] 6 tablas creadas con claves foráneas funcionando
- [ ] Datos insertados en todas las tablas
- [ ] Al menos 5 consultas JOIN ejecutadas correctamente
- [ ] Modificación y eliminación de datos funcionando

Preguntas de cierre:

- ¿Por qué usamos `DECIMAL(3,2)` para notas y no `FLOAT`?
- ¿Qué pasa si intentas borrar una facultad que tiene carreras asociadas?